Contrastive Branch Policy Optimization

Ying Wang[1,2] **Changlin Qiu**[1,*] **Bang Lin**[1] **Linbo Jin**[1] **Wen Jiang**[1] **Zhe Sun**[1] **Jingli Yang**[2]

[1]Alibaba Group, Hangzhou, China [2]Harbin Institute of Technology, Harbin, China

Abstract

Reinforcement learning with verifiable rewards (RLVR) enables language models to learn multi-turn interaction with external tools, yet its sparse outcome rewards provide no signal for identifying which intermediate decisions are responsible for success. Branch sampling induces local comparisons among alternative continuations, but existing methods tend to conflate two distinct problems: allocating a fixed rollout budget and translating branch outcomes into token-level credit. We introduce Contrastive Branch Policy Optimization (CBPO), which disentangles these two problems and assigns a dedicated mechanism to each. Generation entropy screens candidate branch positions across the entire response, while path-level and node-level decay distribute a fixed budget across trajectories and positions to prevent exploration from collapsing onto a few paths or adjacent tokens. A parent trajectory together with the branches that share an identical token prefix forms an exact-prefix group, and the reward variation within this controlled group defines the Contrastive Branch Value (CBV), an outcome-based estimate of local decision sensitivity that rescales continuation advantages without altering their sign. When multiple nodes are selected along the same trajectory, CBPO partitions it into non-overlapping credit segments, thereby avoiding duplicated gradients on shared tokens. Requiring only outcome rewards and no process-level annotation, CBPO provides a practical solution for fine-grained credit assignment in tool-integrated agent training. Extensive experiments on ten benchmarks—five for mathematical reasoning and five for knowledge-intensive search—show that CBPO consistently outperforms state-of-the-art policy-optimization and branch-based methods, attaining the highest macro-average accuracy in both domains and across two model scales.

Keywords

reinforcement learning, large language models, tool-integrated reasoning, credit assignment

1 Introduction

Chain-of-thought prompting [33] and reinforcement learning with verifiable rewards (RLVR) [28, 29] have substantially improved the reasoning capabilities of large language models (LLMs). Many tasks, however, require current information, exact computation, or feedback from an external environment. Addressing such tasks entails multi-turn interaction with search engines, code interpreters, or both [9, 37]. Recent systems apply reinforcement learning to these interactions [7, 20], including settings that require coordinated use of multiple tools [5].

The resulting training signal nevertheless remains coarse. Most RLVR and agentic reinforcement-learning methods assign a single terminal reward to every generated token in a trajectory. Such uniform credit cannot distinguish a decisive tool request or correction from text that merely precedes a successful answer. Process supervision [22] and step-wise preference optimization [19] provide finer signals but require intermediate labels or preferences. Tree-based sampling instead compares continuations from a shared history using only outcome rewards [14]. This approach, however, still requires a principled policy for locating branches under a limited budget and mapping their outcomes to local policy updates.

Existing methods address these requirements only partially. GIGPO and Tree-GRPO construct local advantages from shared or tree-structured histories [8, 16], whereas ARPO uses generation entropy to select branch points after tool feedback [4]. Entropy provides an inexpensive proxy for token-level uncertainty [2, 24, 32], but does not measure outcome sensitivity. Allocation based solely on entropy can concentrate branches at adjacent positions, overlook consequential decisions outside tool boundaries, or emphasize variation that leaves the final answer unchanged.

Contrastive Branch Policy Optimization (CBPO) separates candidate discovery from credit assignment. CBPO scans local entropy throughout the response and applies path-level and node-level decay to distribute a fixed branch budget. A parent trajectory and branches sharing an identical token prefix form an exact-prefix group. Reward variation within this controlled group defines Contrastive Branch Value (CBV), an outcome-based estimate of local decision sensitivity. A bounded form of CBV changes the magnitude, but not the sign, of continuation advantages. Copied prefixes receive no branch gradient, and multiple branch nodes partition a parent trajectory into non-overlapping credit segments. The two signals therefore serve distinct functions: entropy identifies alternative continuations, whereas observed outcome variation estimates their task relevance.

The evaluation spans two model scales and includes five tool-augmented mathematical benchmarks and five knowledge-intensive search benchmarks. CBPO consistently outperforms state-of-the-art policy-optimization and branch-based methods, attaining the highest macro average in both domains.

The main contributions are as follows:

- Branch selection is formulated as budgeted exploration over the entire response. Fixed-interval entropy windows expose candidates beyond tool boundaries, while path-level and node-level decay prevent allocation from collapsing onto a few trajectories or adjacent positions.
- CBV estimates local decision sensitivity from reward variation within exact-prefix groups. Standardized and bounded CBV modulates continuation advantages without changing their signs. Prefix masking and non-overlapping segmentation prevent repeated credit on shared tokens.

*Corresponding author: Changlin Qiu (qiuchanglin.qcl@taobao.com).

- Experiments across ten benchmarks and two model scales show that CBPO consistently outperforms strong policy-optimization and branch-based baselines. Cross-scale ablations further attribute these gains to balanced branch allocation and outcome-contrastive credit assignment.

2 Related Work

2.1 Fine-Grained Credit Assignment

PPO optimizes a policy from trajectory-level returns [28]. GRPO-based reasoning systems likewise derive one advantage from each completed response [10, 29]. Subsequent variants improve optimization stability or scaling [15, 38], while GSPO moves the optimization unit from tokens to sequences [41]. These methods train directly from outcome rewards but do not distinguish decisive intermediate choices from weakly related transitional text. Problem decomposition, computational correction, tool selection, and answer verification can contribute differently to a multi-step solution. Assigning a single advantage may therefore reinforce effective decisions and incidental behavior together.

Process rewards [22] and step-wise preference optimization [19] supervise intermediate reasoning more directly. Preference objectives such as DPO [26] and token-entropy methods [11] offer alternative local signals. Process labels and learned value estimates can provide dense feedback but entail annotation costs or estimation error. Entropy requires neither annotation nor learned value estimation, yet measures predictive uncertainty rather than a position's empirical association with the final reward.

Tree-based methods reuse prefixes and sample several continuations from intermediate states, enabling local comparisons at controlled cost. Online variants derive advantages from current-policy descendants, within-tree contrasts, or recurring histories [14, 16]. Entropy can identify uncertain candidates at low computational cost [2, 24]. Moreover, a small fraction of high-entropy tokens can dominate effective updates [32]. EAPO combines reward polarity with token entropy. AEPO balances entropy during rollout and optimization while limiting repeated branches at consecutive high-entropy positions [6, 11]. These approaches use uncertainty to guide optimization, but divergent token distributions need not yield divergent outcomes. CBPO instead restricts entropy to candidate screening and estimates local credit from outcome contrasts among continuations sharing an identical prefix.

2.2 Tool-Integrated Reasoning

External tools allow language models to access current information, perform exact computation, and act on an environment [23, 37]. Tool-integrated reasoning extends beyond the binary decision to invoke a tool. A model must select a tool, construct the request, determine when to invoke it, verify the response, integrate the result, and terminate appropriately. Search provides open-world evidence, whereas code interpreters support calculation and programmatic verification. Coordinating both requires repeated transitions between internal reasoning and external actions as new observations arrive.

Work in this area has progressed from prompted reasoning and acting [9, 37] to reinforcement learning for search [18, 30], code interpreters [7, 20], and multi-tool coordination [5]. Recent WSDM studies train autonomous programmatic agents and optimize tool selection with reinforcement learning [17, 39]. Unlike prompting or supervised imitation, reinforcement learning can optimize the complete interaction from environmental feedback and terminal outcomes. In long trajectories, however, success may depend on pre-call reasoning, request construction, feedback interpretation, or post-call synthesis. Tool-return boundaries therefore capture only a subset of potentially decisive positions.

Knowledge-intensive tasks also depend on retrieval quality and timing. Adaptive retrieval can extend beyond an initial ranking when the first-stage candidate pool has insufficient recall [27]. In LLM-based fact-checking, adaptive evidence retrieval allows a model to determine when internal knowledge is inadequate or conflicts with external evidence [40]. These WSDM studies concern retrieval and verification rather than token-level credit assignment, but reveal the same structural challenge. An agent must determine where additional computation or evidence can alter the final outcome.

Long-horizon tool learning often improves planning or attribution through reward shaping, process supervision, or step-level optimization. Intermediate supervision is costly, while predefined states and step boundaries constrain the granularity of credit assignment. GIGPO derives step-level advantages from shared histories. ARPO branches after tool feedback and assigns the resulting advantage to the suffix [4, 8]. The former depends on repeated history states, and the latter treats tool returns as fixed branch boundaries. CBPO searches the full response for branch candidates, then attributes local credit by comparing outcomes under an identical prefix.

3 Preliminaries

Tool-integrated rollout. Given a problem x drawn from dataset $\mathcal{D}$ and a tool set $\mathcal{T}$, policy π_θ generates an interaction trajectory τ that interleaves model tokens, tool requests, and environmental observations. Let $z = (z_1, \ldots, z_{|z|})$ collect the model-generated tokens of τ, and let $o_{<t}$ denote the tool observations available before position t. Trajectory generation then factorizes autoregressively over model tokens only:

$$P_\theta(\tau \mid x; \mathcal{T}) = \prod_{t=1}^{|z|} \pi_\theta(z_t \mid x, z_{<t}, o_{<t}; \mathcal{T}). \tag{1}$$

Observations are inserted by the environment upon each tool call; they condition subsequent generation but contribute neither likelihood terms nor policy gradients. Upon termination, a verifier assigns a bounded outcome reward $R(\tau)$. Following agentic reinforcement learning formulations [4, 6], training maximizes the KL-regularized expected reward:

$$\max_{\pi_\theta} \mathbb{E}_{x\sim\mathcal{D},\,\tau\sim\pi_\theta(\cdot|x;\mathcal{T})}\big[R(\tau)\big] - \beta\, D_{\mathrm{KL}}(\pi_\theta \,\|\, \pi_{\mathrm{ref}}), \tag{2}$$

where π_{ref} is a frozen reference policy and β controls the regularization strength. Shared-history, suffix-based, and tree-search agent RL methods adopt this trajectory-level reward setting [4, 8, 16].

Group-relative optimization. For a given problem, Group Relative Policy Optimization (GRPO) samples G trajectories and estimates the advantage of trajectory k as [29]:

$$A_k = \frac{R(\tau_k) - \mu_R}{\sigma_R + \epsilon}, \qquad \mu_R = \frac{1}{G}\sum_{j=1}^{G} R(\tau_j), \tag{3}$$

Here, μ_R and σ_R are the within-group reward mean and standard deviation, and ϵ provides numerical stability. Standard GRPO assigns A_k to every model-generated token in trajectory k, so the update magnitude is uniform along the trajectory regardless of which intermediate decisions determine the outcome.

Generation uncertainty. At generation position t, let the context be $h_t = (x, z_{<t})$ and the next-token distribution be $p_t = \pi_\theta(\cdot \mid h_t)$. Its Shannon entropy is

$$H_t = -\sum_{v \in \mathcal{V}} p_{t,v} \log p_{t,v}, \tag{4}$$

where $\mathcal{V}$ denotes the vocabulary. Larger H_t indicates greater uncertainty over the next token. This quantity reflects the dispersion of the generation distribution at position t rather than the confidence of any particular sampled token. Prior work uses token entropy to identify critical decisions [24] and characterize exploration in language-model reasoning [2, 32].

4 Contrastive Branch Policy Optimization

4.1 Overview

CBPO addresses two coupled limitations of branch-based RL: concentrated exploration and imprecise local credit. The method uses generation uncertainty for candidate selection and observed outcome variation to modulate update magnitude. Figure 1 summarizes the resulting training framework. Starting from complete interaction trajectories, CBPO scans each response for uncertain positions and allocates a fixed branch budget through path-level and node-level decay. Rewards are then compared among continuations sharing an exact prefix to construct CBV-aware local credit. Shared-prefix deduplication and non-overlapping segmentation convert this credit into token-level advantages. Section 4.2 constructs balanced exact-prefix groups, and Section 4.3 converts their outcomes into policy updates.

The design maintains three invariants. First, every problem receives the same total rollout budget because unused branch slots are reallocated to independent complete trajectories. Second, each local comparison conditions on an identical token history, so reward variation arises only from resampled continuations. Third, copied prefixes are excluded from branch losses, and parent trajectories are partitioned into non-overlapping intervals when several nodes are selected. In tool-integrated trajectories, environmental observations condition later actions but are neither scanned nor assigned policy gradients, since they are not model-generated. These constraints separate allocation from attribution and prevent additional branches from duplicating shared-token gradients.

4.2 Entropy-Guided Balanced Branch Exploration

Exhaustive branching at every token is computationally infeasible, whereas branching only at tool-return boundaries assumes that consequential decisions coincide with predefined interaction events. Entropy provides an inexpensive candidate signal, but an unregularized ranking can allocate most of a fixed budget to a few trajectories or neighboring positions. This stage therefore combines full-response candidate discovery with explicit coverage control. The procedure first samples N_0 complete trajectories from the current policy. Figure 2 illustrates how fixed-interval scanning identifies high-entropy decisions beyond tool-call boundaries.

Full-trajectory candidate identification. For trajectory i, CBPO places candidate boundaries throughout the response at intervals of $d_{\min}$:

$$\mathcal{B}_i = \{b_{i,q} = q d_{\min}\}_{q=1}^{Q_i}, \tag{5}$$

where Q_i is the number of valid candidates. Let $\mathcal{W}_{i,b}$ denote the window of w model-generated tokens beginning at boundary $b \in \mathcal{B}_i$. Its normalized entropy is

$$H_{i,b} = -\frac{1}{|\mathcal{W}_{i,b}| \log |\mathcal{V}|} \sum_{t \in \mathcal{W}_{i,b}} \sum_{v \in \mathcal{V}_K^{i,t}} p_{i,t,v} \log p_{i,t,v}, \tag{6}$$

where $\mathcal{V}_K^{i,t}$ is the top-K candidate set at token t, and $p_{i,t,v}$ is the probability of candidate v. The retained probability mass is not renormalized. Thus, $H_{i,b}$ is a truncated entropy proxy used only to rank candidate locations, not the entropy of a top-K distribution.

The first window of each trajectory provides a path-specific reference:

$$H_i^{\text{root}} = H_{i,0}. \tag{7}$$

Local uncertainty at boundary b is measured by the entropy increase relative to this reference:

$$\Delta H_{i,b} = H_{i,b} - H_i^{\text{root}}. \tag{8}$$

The corresponding raw priority is

$$P_{i,b}^{\text{raw}} = \text{clip}\left(\alpha + \gamma \Delta H_{i,b}, 0, 1\right), \tag{9}$$

where α sets the base priority, γ scales the entropy increase, and clip restricts the score to $[0, 1]$.

Path-node balanced budgeting. An entropy ranking alone can allocate most branches to a few trajectories or neighboring positions. Let L_i count branches assigned to path i, and let $l_{i,b}$ count branches at node (i, b). CBPO updates the priority as

$$P_{i,b}^{\text{bal}} = \frac{P_{i,b}^{\text{raw}}}{(1 + L_i)^{\rho_{\text{path}}} (1 + l_{i,b})^{\rho_{\text{node}}}}, \tag{10}$$

where $\rho_{\text{path}}, \rho_{\text{node}} \geq 0$ control path-level and node-level decay. Before allocation, CBPO retains at most $J_{\max}$ candidates from each parent. A candidate remains eligible while $l_{i,b} < B_{\text{node}}$ and $L_i < B_{\text{path}}$. The hard caps are necessary because power decay never reduces a priority exactly to zero. Updating both counts after each sampled branch progressively shifts allocation toward less-explored paths and nodes.

At each allocation step, CBPO selects the eligible candidate with the largest P^{bal}. A branch is sampled only if this score exceeds κ. Otherwise, independent complete trajectories fill the unused rollout slots, preserving the fixed budget.

At a selected node, the trajectory is decomposed at token boundary b as

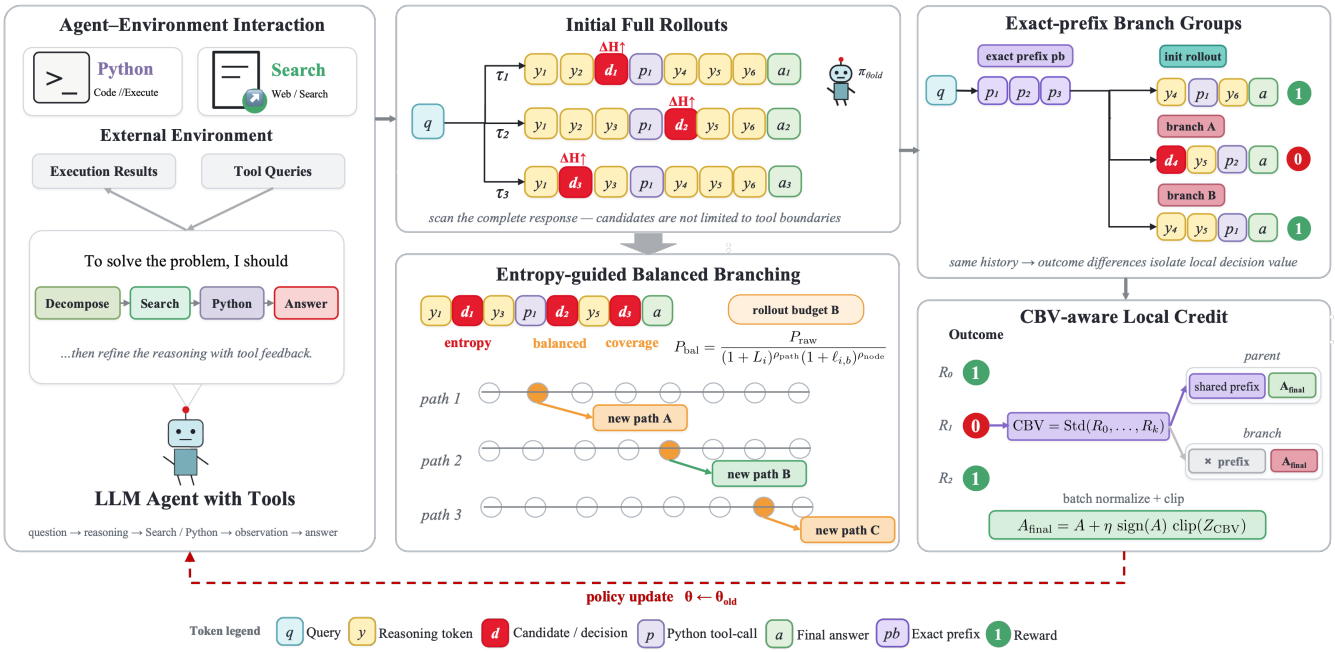

Figure 1: CBPO training framework. The model first performs interactive reasoning with Python and Search tools and samples complete initial trajectories. Candidate nodes are identified across the full response, and path-level and node-level decay allocate the branch budget. Outcome rewards estimate CBV within exact-prefix groups. Shared-prefix deduplication and non-overlapping hierarchical segmentation then construct token-level advantages for the policy update.

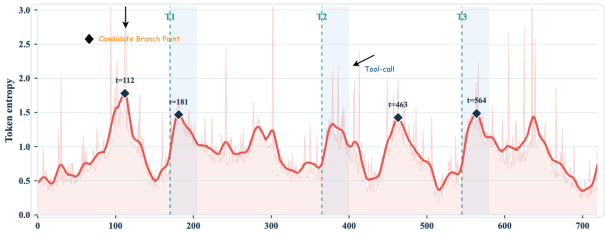

Figure 2: Full-response candidate discovery. Fixed-interval entropy scans identify high-entropy decisions beyond predefined tool-call boundaries.

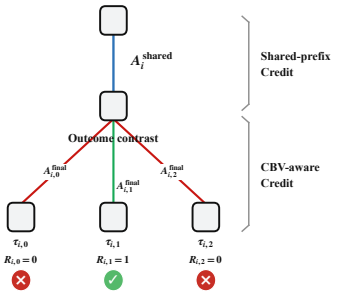

Figure 3: Outcome-contrastive credit assignment. Exact-prefix continuations are compared by outcome to construct CBV-aware credit while deduplicating shared-prefix credit.

$$\tau = (p_b, c_b), \qquad p_b = \tau_{<b}, \qquad c_b = \tau_{\geq b}, \tag{11}$$

Branch generation holds prefix p_b fixed and resamples only continuation c_b. The parent and all branches sharing p_b form an exact-prefix group $\mathcal{G}(p_b)$. Comparing their outcomes controls for the preceding token history and isolates variation among the sampled continuations. The resulting balanced exact-prefix groups serve as input to the outcome-contrastive credit-assignment stage.

4.3 Hierarchical Policy Optimization with CBV

The first stage identifies positions at which the policy admits alternatives, but entropy alone cannot establish whether those alternatives affect the reward. The second stage therefore measures reward variation within each exact-prefix group and translates it into local credit. This formulation emphasizes continuations sampled at outcome-sensitive nodes while suppressing redundant gradients on their shared histories. Figure 3 illustrates the distinction between shared-prefix credit and CBV-aware continuation credit.

CBPO requires only bounded outcome rewards. Following ReTool and ToRL [7, 20], mathematical tasks use a binary correctness reward that compares trajectory answer $\hat{a}(\tau)$ with reference answer $a^\star$:

$$R(\tau) = \begin{cases} 1, & \hat{a}(\tau) \text{ is symbolically equivalent to } a^\star, \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

Search tasks use normalized token-level F1 or an LLM-as-a-Judge score. Every reward satisfies $R(\tau) \in [0, 1]$; Section 5.1 gives the benchmark-specific metrics.

Contrastive Branch Value. Consider exact-prefix group $\mathcal{G}_i = \{\tau_{i,0}, \ldots, \tau_{i,K_i}\}$, where $\tau_{i,0}$ is the parent and the remaining trajectories are branches. Contrastive Branch Value (CBV) is defined as the population standard deviation of group rewards $R_{i,0}, \ldots, R_{i,K_i}$:

$$\mathrm{CBV}_i = \sqrt{\frac{1}{K_i+1}\sum_{k=0}^{K_i}\left(R_{i,k} - \bar{R}_i\right)^2}, \qquad \bar{R}_i = \frac{1}{K_i+1}\sum_{k=0}^{K_i} R_{i,k}. \quad (13)$$

Larger CBV indicates greater reward variation among continuations sampled after the same prefix. Because raw CBV is nonnegative, direct addition would only increase advantage values. Standardization across valid nodes in the batch produces a comparable signed signal:

$$Z_i^{\mathrm{CBV}} = \frac{\mathrm{CBV}_i - \mu_{\mathrm{CBV}}}{\sigma_{\mathrm{CBV}} + \epsilon}. \quad (14)$$

Here, $Z_i^{\mathrm{CBV}} > 0$ denotes reward variation above the batch mean. If σ_{CBV} falls below a numerical threshold, every Z_i^{CBV} is set to zero and the base advantages are retained.

Credit assignment. CBV identifies relative outcome sensitivity at the group level, but the optimizer still requires token-level advantages. Problem-level GRPO normalization first yields base advantage $A_{i,k}$ for each trajectory. The common prefix at node i receives the group-mean advantage:

$$A_i^{\mathrm{shared}} = \frac{1}{K_i+1}\sum_{k=0}^{K_i} A_{i,k}. \quad (15)$$

Following bounded advantage modulation [11], CBV changes only the magnitude of continuation advantage k. The standardized credit is bounded as

$$C_{i,k}^{\mathrm{CBV}} = \mathrm{clip}\left(Z_i^{\mathrm{CBV}}, -\frac{|A_{i,k}|}{\phi}, \frac{|A_{i,k}|}{\phi}\right). \quad (16)$$

The CBV-aware advantage is

$$A_{i,k}^{\mathrm{final}} = A_{i,k} + \eta\ \mathrm{sign}(A_{i,k})\ \mathrm{stopgrad}\left(C_{i,k}^{\mathrm{CBV}}\right), \quad (17)$$

where η controls modulation strength and ϕ limits its relative magnitude. For $0 \le \eta < \phi$, the modulated advantage retains the original sign and therefore preserves the local optimization direction.

Hierarchical segmentation for multiple nodes. Let parent trajectory i contain J_i selected boundaries

$$b_{i,1} < b_{i,2} < \cdots < b_{i,J_i}. \quad (18)$$

Multiple selected ancestors can otherwise assign credit repeatedly to the same suffix. CBPO therefore partitions each parent trajectory into non-overlapping intervals, with token advantage

$$\widetilde{A}_t = \begin{cases} A_{i,1}^{\mathrm{shared}}, & 0 \le t < b_{i,1}, \\ \frac{1}{2}\left(A_{i,j,0}^{\mathrm{final}} + A_{i,j+1}^{\mathrm{shared}}\right), & b_{i,j} \le t < b_{i,j+1}, \\ A_{i,J_i,0}^{\mathrm{final}}, & t \ge b_{i,J_i}. \end{cases} \quad (19)$$

where $A_{i,j,0}^{\mathrm{final}}$ is the parent's continuation advantage at its j-th selected node. Each intermediate interval is simultaneously a suffix of the preceding node and a shared prefix of the following node. The two associated terms therefore receive equal weight.

CBPO retains the GRPO objective, replacing only trajectory-level advantage A_k with segmented token advantage $\widetilde{A}_{k,t}$. For model-generated token $y_{k,t}$, the importance ratio is

$$r_{k,t}(\theta) = \frac{\pi_\theta(y_{k,t} \mid x, y_{k,<t})}{\pi_{\mathrm{old}}(y_{k,t} \mid x, y_{k,<t})}, \quad (20)$$

and its clipped form

$$\bar{r}_{k,t}(\theta) = \mathrm{clip}\big(r_{k,t}(\theta), 1-\varepsilon, 1+\varepsilon\big), \quad (21)$$

$$\ell_{k,t}(\theta) = \min\Big(r_{k,t}(\theta)\widetilde{A}_{k,t}, \bar{r}_{k,t}(\theta)\widetilde{A}_{k,t}\Big). \quad (22)$$

The CBPO objective is

$$\mathcal{J}_{\mathrm{CBPO}}(\theta) = \mathbb{E}\left[\frac{1}{M}\sum_{k=1}^{M}\frac{1}{|y_k|}\sum_{t=1}^{|y_k|}\left(\ell_{k,t}(\theta) - \beta D_{\mathrm{KL}}\left(\pi_\theta \,\|\, \pi_{\mathrm{ref}}\right)\right)\right]. \quad (23)$$

The objective connects the two stages: balanced exploration defines the controlled comparisons, and CBV determines their contribution to non-overlapping continuation updates. Algorithm 1 summarizes the end-to-end training procedure.

Algorithm 1 Contrastive Branch Policy Optimization

Input initial policy $\pi_{\theta_{\mathrm{init}}}$, reference policy π_{ref}, data $\mathcal{D}$, tools $\mathcal{T}$
Input training steps S, rollout budget M, initial count N_0, caps $J_{\max}, B_{\mathrm{node}}, B_{\mathrm{path}}$
Input $\alpha, \gamma, \kappa, \rho_{\mathrm{path}}, \rho_{\mathrm{node}}, d_{\min}, w, K, \eta, \phi$
Output trained policy π_θ

$\pi_\theta \leftarrow \pi_{\theta_{\mathrm{init}}}$
for training step $s = 1, \ldots, S$ **do**
  $\pi_{\mathrm{old}} \leftarrow \pi_\theta$; sample minibatch $\mathcal{D}_b \subset \mathcal{D}$
  for all problems $x \in \mathcal{D}_b$ **do**
    sample N_0 parent trajectories; initialize rollout set $\mathcal{R}_x$
    for all parent trajectories $\tau_i \in \mathcal{R}_x$ **do**
      scan $d_{\min}$-spaced boundaries and compute $H_{i,b}, \Delta H_{i,b}, P_{i,b}^{\mathrm{raw}}$
      retain the top $J_{\max}$ candidates; set $L_i \leftarrow 0$ and $l_{i,b} \leftarrow 0$
    while $|\mathcal{R}_x| < M$ **do**
      form eligible set $C_x = \{(i,b) : L_i < B_{\mathrm{path}},\ l_{i,b} < B_{\mathrm{node}}\}$
      if $C_x = \varnothing$ **then**
        sample $M - |\mathcal{R}_x|$ independent complete trajectories; **break**
      compute $P_{i,b}^{\mathrm{bal}}$ and choose $(i^*, b^*) = \arg\max_{(i,b)\in C_x} P_{i,b}^{\mathrm{bal}}$
      if $P_{i^*,b^*}^{\mathrm{bal}} > \kappa$ **then**
        fix prefix $\tau_{i^*,<b^*}$, resample one continuation, and add it to $\mathcal{R}_x$
        $L_{i^*} \leftarrow L_{i^*} + 1$; $l_{i^*,b^*} \leftarrow l_{i^*,b^*} + 1$
      else
        sample $M - |\mathcal{R}_x|$ independent complete trajectories; **break**
    evaluate $R(\tau)$ and compute prompt-level base advantages $A_{i,k}$ over $\mathcal{R}_x$
  build exact-prefix groups; compute CBV_i, Z_i^{CBV}, A_i^{shared}, and $A_{i,k}^{\mathrm{final}}$
  mask each copied branch prefix; assign $A_{i,k}^{\mathrm{final}}$ only to its sampled suffix
  segment each parent by Eq. (19); keep base advantages for fallback rollouts
  maximize Eq. (23) and update θ

4.4 Theoretical Analysis

Two properties clarify the roles of the two signals. Conditional entropy bounds the outcome information available in sampled continuations, which supports its use for candidate screening. CBV estimates conditional outcome variance under a fixed prefix, which supports its use for local credit. The bounded modulation additionally preserves the advantage sign and local PPO gradient direction.

Property 1: generation entropy bounds attainable outcome information. Given exact prefix p, let continuation

$C = (Y_1, \ldots, Y_L) \sim \pi_\theta(\cdot \mid p)$ produce final outcome reward R. The chain rule for conditional entropy and the upper bound on conditional mutual information give

$$H(C \mid p) = \sum_{t=1}^{L} \mathbb{E}_{Y_{<t}|p}\big[H(Y_t \mid p, Y_{<t})\big], \qquad I(C; R \mid p) \le H(C \mid p). \tag{24}$$

For deterministic outcome verification, a nearly deterministic continuation can carry little information about alternative outcomes. High conditional entropy is not sufficient, however, because variation in wording, formatting, or inconsequential reasoning may leave the answer unchanged. Thus, entropy provides an upper bound rather than a direct estimate of $I(C; R \mid p)$. CBPO uses it to screen for positions where meaningful alternatives may exist, then relies on observed branch rewards for credit assignment.

Property 2: CBV estimates conditional outcome variance under a shared prefix. Suppose n continuations are sampled independently from prefix p, yielding rewards $R_1, \ldots, R_n$, and let $\bar{R} = n^{-1} \sum_k R_k$. Then

$$\mathrm{CBV}^2(p) = \frac{1}{n} \sum_{k=1}^{n} (R_k - \bar{R})^2 = \frac{1}{2n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} (R_i - R_j)^2, \tag{25}$$

and

$$\mathbb{E}\big[\mathrm{CBV}^2(p) \mid p\big] = \frac{n-1}{n} \mathrm{Var}(R \mid p). \tag{26}$$

The pairwise identity follows by expanding the squared differences:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} (R_i - R_j)^2 = 2n \sum_{i=1}^{n} R_i^2 - 2 \left(\sum_{i=1}^{n} R_i \right)^2. \tag{27}$$

For $i \neq j$, conditional independence gives $\mathbb{E}[(R_i - R_j)^2 \mid p] = 2\,\mathrm{Var}(R \mid p)$. The n diagonal terms are zero, leaving $n(n-1)$ nonzero ordered pairs. Substitution into Eq. (25) yields Eq. (26).

Equations (25) and (26) show that $\frac{n}{n-1}\,\mathrm{CBV}^2(p)$ is an unbiased estimator of conditional outcome variance. Its pairwise form measures reward divergence among continuations sharing the same token history. For binary correctness with $q_p = \Pr(R = 1 \mid p)$, this variance is $q_p(1 - q_p)$ and is maximal at $q_p = 1/2$. Consequently, an uncertain node with consistent outcomes receives little CBV, whereas a node separating success from failure receives more. Bounded modulation changes update magnitude within a controlled interval while preserving the advantage sign and local PPO gradient direction.

Property 3: bounded CBV modulation preserves the update direction. Let $C = C^{\mathrm{CBV}}$ satisfy the clipping constraint $|C| \le |A|/\phi$. For any nonzero base advantage A, Eq. (17) can be written as

$$A^{\mathrm{final}} = wA, \qquad w = 1 + \eta \frac{C}{|A|} \in \left[1 - \frac{\eta}{\phi}, 1 + \frac{\eta}{\phi}\right]. \tag{28}$$

When $0 \le \eta < \phi$, the lower endpoint is positive. Therefore, A^{final} has the same sign as A. The PPO surrogate is positively homogeneous in its advantage argument, so $\ell(r, wA) = w\ell(r, A)$ for $w > 0$. Because the modulation is stop-gradient, its local policy gradient is also scaled by the same positive w. CBV consequently reallocates update magnitude without reversing the preference induced by the outcome reward. When $A = 0$, the clipping constraint forces $C = 0$, and the statement holds trivially.

5 Experiments

5.1 Experimental Setup

Datasets and evaluation. The evaluation covers mathematical reasoning and knowledge-intensive search, two forms of tool-integrated reasoning. Each task provides access to Web Search and Python, allowing the policy to select tools without dataset-specific routing. Mathematical evaluation uses AIME 2024, AIME 2025, MATH-500, MATH [12], and the complete GSM8K test set [3]. Reported metrics include Pass@1, the five-task macro average, and Pass@3/5 computed from five samples per problem. Knowledge-intensive search evaluation uses WebWalker [34], HotpotQA [36], 2WikiMultiHopQA [13], MuSiQue [31], and Bamboogle [25]. WebWalker is evaluated with LLM-as-a-Judge Pass@1, whereas the other four benchmarks use normalized token-level F1. Domain-specific averages are reported because these metrics are not directly comparable.

Evaluation controls. All methods use the same system prompt, tool interfaces, decoding settings, and answer-extraction procedure. Search calls return the top ten snippets and share an evaluation cache, preventing method-specific retrieval variation from confounding policy comparisons. Mathematical answers are checked with the same symbolic-equivalence verifier. Pass@3/5 uses five independent samples for every method and problem. AIME 2024 and AIME 2025 each contain 30 problems, so one additional correct answer changes accuracy by 3.3 percentage points. Accordingly, the analysis emphasizes cross-task averages, both model scales, and component ablations rather than small AIME differences in isolation.

Training setup. The experiments use Qwen3-1.7B and Qwen3-4B backbones [35]. Training data are drawn from Tool-Star [5]. Mathematical experiments use 5,000 mathematical samples, whereas search experiments use search and dual-tool trajectories from 10,000 open-domain RL samples. All reward-based methods share the SFT initialization, training data, update count, outcome rewards, tool environment, and 16 rollout slots. OPD checkpoints follow the same benchmark and decoding protocol during evaluation. CBPO allocates six slots to initial trajectories and ten to a second branching stage. Independent complete trajectories fill any unused branch slots. Candidate scanning uses $w = 20$, $K = 10$, $d_{\min} = 64$, and $J_{\max} = 3$. Allocation uses $\alpha = 0.2$, $\gamma = 2.0$, $\kappa = 0.25$, $\rho_{\mathrm{path}} = \rho_{\mathrm{node}} = 0.2$, $B_{\mathrm{node}} = 3$, and $B_{\mathrm{path}} = 4$. Credit modulation uses $\eta = 0.2$, $\phi = 2.0$, and a CBV standard-deviation threshold of 10^{-6}.

Baselines. At both model scales, the comparison includes Base, TIR prompting [23], SFT, and on-policy distillation (OPD) [21]. Policy-optimization baselines are GRPO [29], REINFORCE++ [15], DAPO [38], GSPO [41], EAPO [11], and OC-GRPO [1]. Agentic and branch-based baselines are GIGPO [8], Tree-GRPO [16], and ARPO [4].

Table 1: Measured results (%) for Qwen3-1.7B and Qwen3-4B under a unified protocol. Mathematical tasks report Pass@1; WebWalker reports LLM-as-a-Judge Pass@1; the other search tasks report normalized token-level F1. Math Avg. and Search Avg. are unweighted five-task means. Green subscripts show improvement over size-matched GIGPO.

Method	Mathematical Reasoning						Knowledge-Intensive Search					
	AIME24	AIME25	MATH500	GSM8K	MATH	Math Avg.	WebWalker	HotpotQA	2Wiki	MuSiQue	Bamboogle	Search Avg.
Backbone: Qwen3-1.7B												
Training-Free Method												
Base	13.3	16.7	82.2	90.8	86.7	57.9	5.0	22.0	25.0	9.5	34.0	19.1
TIR Prompting	16.7	23.3	81.4	88.4	85.1	59.0	14.5	36.0	41.0	16.0	48.0	31.1
Supervised and Distillation Methods												
SFT	16.7	23.3	82.8	91.0	87.6	60.3	16.5	48.0	58.0	22.5	58.0	40.6
OPD	20.0	26.7	84.4	92.0	88.7	62.4	18.5	52.0	66.5	25.0	60.5	44.5
Classic RL Method												
GRPO	16.7	30.0	83.6	91.3	89.0	62.1	18.0	53.5	68.5	26.5	62.0	45.7
REINFORCE++	20.0	26.7	83.2	91.9	88.3	62.0	18.5	51.0	65.5	24.0	61.0	44.0
DAPO	13.3	26.7	84.8	91.5	88.8	61.0	17.5	52.0	65.0	25.0	60.5	44.0
GSPO	20.0	26.7	84.6	91.0	88.7	62.2	18.5	54.0	68.0	26.0	63.0	45.9
EAPO	23.3	30.0	85.6	91.8	90.0	64.1	20.5	55.0	69.5	**32.0**	65.0	48.4
OC-GRPO	20.0	**33.3**	85.2	92.4	89.5	64.1	19.5	54.5	69.0	27.0	64.0	46.8
Agentic RL Method												
GIGPO	23.3	26.7	84.2	92.4	89.1	$63.1_{\Delta_{\text{base}}}$	23.5	58.0	73.0	29.5	67.0	$50.2_{\Delta_{\text{base}}}$
Tree-GRPO	23.3	30.0	85.0	92.0	88.8	$63.8_{+1.1\%}$	24.0	58.5	73.5	30.0	68.0	$50.8_{+1.2\%}$
ARPO	**26.7**	30.0	85.0	91.7	88.3	$64.3_{+1.9\%}$	24.5	59.0	74.0	30.5	69.0	$51.4_{+2.4\%}$
CBPO	**26.7**	**33.3**	**86.4**	**93.2**	**90.4**	$\mathbf{66.0}_{+4.6\%}$	**26.0**	**61.0**	**75.5**	31.5	**72.0**	$\mathbf{53.2}_{+6.0\%}$
Backbone: Qwen3-4B												
Training-Free Method												
Base	13.3	20.0	84.6	90.0	89.8	59.5	7.0	27.0	31.0	12.0	40.0	23.4
TIR Prompting	16.7	26.7	85.1	92.7	89.3	62.1	17.5	41.5	47.0	19.5	53.5	35.8
Supervised and Distillation Methods												
SFT	16.7	26.7	85.7	93.2	90.5	62.6	20.0	52.0	63.0	25.5	63.0	44.7
OPD	23.3	33.3	86.5	94.1	89.6	65.4	21.5	56.0	71.5	28.0	65.5	48.5
Classic RL Method												
GRPO	16.7	**40.0**	86.0	93.8	91.3	65.6	21.5	57.5	73.0	29.5	66.5	49.6
REINFORCE++	26.7	33.3	85.8	94.2	91.4	66.3	22.0	55.0	70.0	27.0	65.5	47.9
DAPO	13.3	30.0	87.8	93.5	91.5	63.2	21.0	56.0	69.0	28.0	65.0	47.8
GSPO	20.0	33.3	87.1	93.4	91.2	65.0	21.5	57.0	72.0	29.0	66.0	49.1
EAPO	23.3	36.7	87.2	94.4	91.8	66.7	24.0	59.5	74.5	**35.0**	69.0	52.4
OC-GRPO	23.3	33.3	87.0	94.2	90.6	65.7	22.5	58.5	73.5	29.5	67.5	50.3
Agentic RL Method												
GIGPO	26.7	33.3	86.7	94.6	91.0	$66.5_{\Delta_{\text{base}}}$	27.0	62.0	77.0	32.5	73.0	$54.3_{\Delta_{\text{base}}}$
Tree-GRPO	26.7	33.3	87.5	94.0	91.9	$66.7_{+0.3\%}$	28.0	62.5	77.5	33.0	74.0	$55.0_{+1.3\%}$
ARPO	**30.0**	33.3	87.4	94.1	90.7	$67.1_{+0.9\%}$	29.0	63.0	78.0	33.5	75.0	$55.7_{+2.6\%}$
CBPO	**30.0**	36.7	**89.5**	**96.3**	**94.0**	$\mathbf{69.3}_{+4.2\%}$	**31.0**	**65.5**	**80.0**	34.5	**76.5**	$\mathbf{57.5}_{+5.9\%}$

5.2 Main Results

The primary comparison evaluates whether the complete CBPO pipeline improves closed-form reasoning and open-world retrieval under matched rollout budgets. Table 1 reports mathematical macro-average Pass@1 scores of 66.0% and 69.3% with the 1.7B and 4B backbones. These scores exceed those of ARPO, the strongest size-matched baseline by macro average, by 1.7 and 2.2 percentage points. The corresponding search averages are 53.2 and 57.5, exceeding ARPO by 1.8 points at both scales. Relative to OPD, CBPO gains 3.6 and 3.9 points on the mathematical average and 8.7 and 9.0 points on the search average. CBPO ranks first or ties for first on all five mathematical benchmarks at 1.7B and on four of five at 4B. At 4B, GRPO leads AIME 2025 by one problem. CBPO also leads four of five search benchmarks at each scale, trailing EAPO on MuSiQue by 0.5 points. This consistency across domains and model scales supports the overall pipeline. The following analyses examine the component-specific hypotheses.

Pass@K extension. Figure 4 evaluates whether the gain persists when several candidate answers are available. From five independent samples per problem, Qwen3-1.7B with CBPO reaches macro-average Pass@1, Pass@3, and Pass@5 scores of 66.0%, 74.6%, and 77.9%. The corresponding Qwen3-4B scores are 69.3%, 76.9%, and 80.2%. CBPO exceeds ARPO at each reported value, indicating that its advantage is not confined to single-sample decoding.

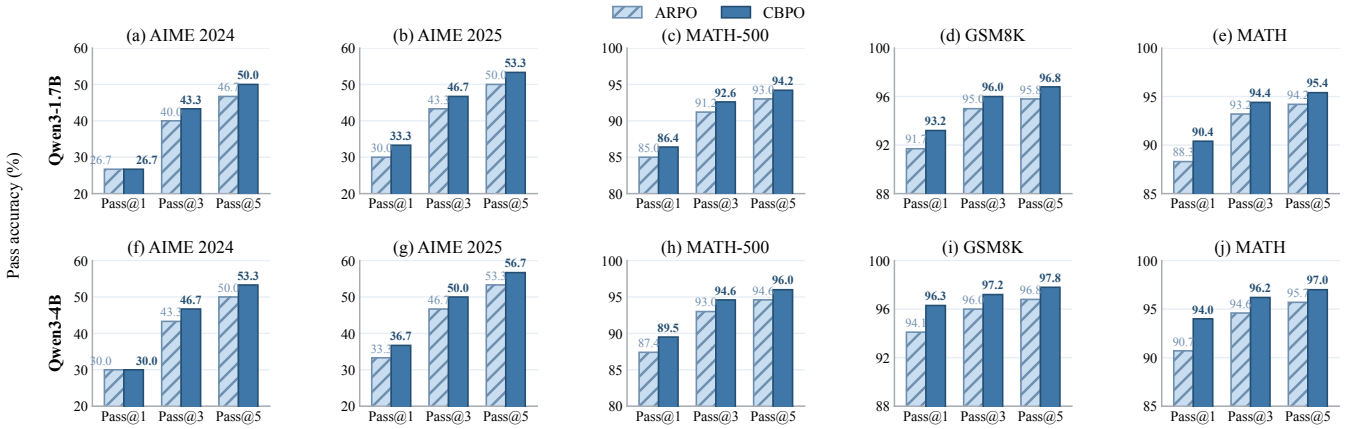

Figure 4: Pass@K comparison between ARPO and CBPO. The top and bottom rows show Qwen3-1.7B and Qwen3-4B. Pass@3/5 is computed from five samples per problem. Each benchmark uses an independent vertical axis; consequently, bar heights should not be compared across benchmarks.

Training dynamics and branch selection. Figure 5 compares reward, tool use, and selected branch positions. CBPO attains a higher final mean reward than GIGPO and ARPO while averaging fewer tool calls. The performance gain therefore cannot be attributed to more frequent tool invocation. The qualitative examples show that decay can reorder high-entropy candidates before allocation, consistent with separating candidate discovery from budget control.

5.3 Full-Response Candidate Analysis

Three Qwen3-1.7B trajectories from the MATH test set were inspected to locate decisions associated with their outcomes. In one successful trajectory, the model enumerated all 16 one- and two-digit candidates before calling Python. The interpreter then correctly identified the ten primes. In a failed trajectory, the submitted code also executed correctly, but the preceding reasoning listed only six parenthesizations of an arithmetic expression. Python evaluated this incomplete list and returned two values instead of the correct four. The decisive error therefore occurred in the reasoning before tool invocation, not at the tool boundary.

The third trajectory exhibited the converse pattern. Python raised the same execution exception twice, yet the model recovered the correct answer from an algebraic derivation completed before either call. A tool-return position was therefore salient but not decisive for the final reward. Together, these cases illustrate why candidate discovery should cover the full response rather than treat tool observations as universal branch boundaries. These examples provide qualitative mechanism checks, not population-level or causal evidence. A controlled tool-boundary-only ablation remains necessary to quantify this design choice independently.

5.4 Ablation Study

The method motivates two quantitative predictions. Path-level and node-level decay should reduce complementary forms of allocation concentration, whereas CBV should improve learning beyond

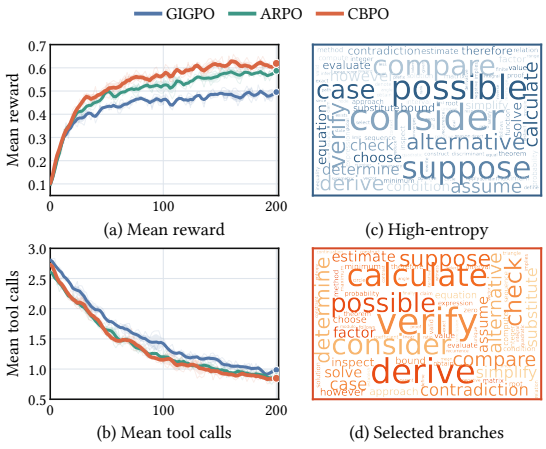

Figure 5: Training dynamics and branch selection. (a) Mean reward and (b) mean tool calls per trajectory; thick lines denote smoothed means, with light traces and shading showing variation. In the qualitative word clouds, (c) token size increases monotonically with candidate entropy, whereas (d) token size increases monotonically with post-decay branching priority among retained candidates. Entropy therefore favors selection, but path- and node-level decay can change the ranking and retain some positions below the entropy-only cutoff; sizes do not encode token frequency.

branch exploration alone. Table 2 tests these predictions under matched training, rollout, and evaluation protocols.

Removing either decay level reduces the mathematical macro average by 1.5 points at both model scales. Removing both reduces it by 2.2 points, more than either individual ablation. The two decay terms therefore regulate distinct sources of allocation concentration. Removing CBV produces the largest losses, 3.0 points for Qwen3-1.7B and 2.9 points for Qwen3-4B. Branch exploration alone is therefore insufficient; branch outcomes must also inform

local credit. Most of this reduction occurs on AIME 2024 and AIME 2025, the two benchmarks with the lowest absolute accuracy.

Table 2: Component ablations for CBPO. Values are Pass@1 (%) and the five-task macro average. Shading marks the full method, and bold indicates the best value in each column.

Method	AIME24	AIME25	MATH500	GSM8K	MATH	Avg.
Backbone: Qwen3-1.7B						
CBPO	**26.7**	**33.3**	**86.4**	**93.2**	**90.4**	**66.0**
w/o Path-level	23.3	30.0	86.2	92.8	90.2	64.5
w/o Node-level	23.3	30.0	86.1	93.0	90.3	64.5
w/o Dual decay	20.0	30.0	86.3	92.5	90.0	63.8
w/o CBV	20.0	26.7	85.5	92.9	90.1	63.0
Backbone: Qwen3-4B						
CBPO	**30.0**	**36.7**	**89.5**	**96.3**	**94.0**	**69.3**
w/o Path-level	26.7	33.3	89.3	95.9	93.8	67.8
w/o Node-level	26.7	33.3	89.0	96.0	93.9	67.8
w/o Dual decay	23.3	33.3	89.4	95.7	93.9	67.1
w/o CBV	23.3	30.0	89.0	96.0	93.7	66.4

Table 3: CBPO Pass@1 (%) under different rollout configurations. $M = N + B$. Shading marks the default configuration, and bold indicates the best value in each column.

Rollout allocation			Pass@1 accuracy (%)					
Total M	Initial N	Branch B	AIME24	AIME25	MATH500	GSM8K	MATH	Avg.
Backbone: Qwen3-1.7B								
4	1	3	13.3	23.3	82.2	89.9	86.0	58.9
4	2	2	16.7	23.3	82.8	90.5	86.8	60.0
8	2	6	16.7	26.7	83.7	91.0	87.8	61.2
8	4	4	20.0	26.7	84.0	91.5	88.3	62.1
16	4	12	23.3	**36.7**	85.8	92.9	89.8	65.7
16	**6**	**10**	**26.7**	33.3	**86.4**	**93.2**	**90.4**	**66.0**
16	8	8	**26.7**	30.0	86.1	93.0	90.2	65.2
16	12	4	23.3	30.0	85.8	92.6	89.8	64.3
Backbone: Qwen3-4B								
4	1	3	16.7	26.7	84.4	91.8	89.0	61.7
4	2	2	20.0	26.7	85.0	92.3	89.7	62.7
8	2	6	20.0	30.0	86.6	93.8	91.0	64.3
8	4	4	23.3	30.0	87.0	94.2	91.5	65.2
16	4	12	26.7	**40.0**	88.9	95.8	93.3	68.9
16	**6**	**10**	**30.0**	36.7	**89.5**	**96.3**	94.0	**69.3**
16	8	8	**30.0**	33.3	89.2	96.0	**94.4**	68.6
16	12	4	26.7	33.3	88.8	95.6	93.2	67.5

5.5 Branching Configuration

Balanced branching requires sufficient independent paths for global coverage and sufficient shared-prefix continuations for local comparison. This trade-off is evaluated by varying the allocation between initial trajectories N and dynamic branches B. Total budgets are $M = N + B \in \{4, 8, 16\}$, with all other settings held fixed.

Increasing M from 4 to 16 improves the macro average at both model scales. Within a fixed budget, too few initial trajectories restrict path coverage, whereas too few branches weaken exact-prefix comparisons. The highest macro average occurs at $N = 6, B = 10$ for both backbones. This shared optimum favors branch sampling while retaining sufficient independent trajectories for global coverage.

6 Conclusion

CBPO addresses fine-grained credit assignment by separating two operations that branch-based RL often conflates. Full-response entropy scanning with path-level and node-level decay allocates a fixed rollout budget. Reward variation within exact-prefix groups then modulates the contribution of each sampled continuation. Prefix masking and hierarchical segmentation prevent shared tokens from receiving duplicated credit. With Qwen3-1.7B and Qwen3-4B, CBPO attains mathematical macro averages of 66.0% and 69.3% and search averages of 53.2 and 57.5. Cross-scale ablations attribute distinct gains to balanced allocation and CBV. However, the independent contribution of full-response scanning still requires a controlled boundary-only ablation. Within the evaluated models, tasks, and budgets, the evidence supports a bounded design principle: uncertainty identifies alternatives, whereas observed outcome variation provides a stronger signal for local credit assignment.

Ethical Considerations

The study uses public benchmarks and collects no new personal or human-subject data. Open-web retrieval may nevertheless expose a model to inaccurate, biased, offensive, or privacy-sensitive material. Benchmark accuracy does not establish the reliability or social neutrality of retrieved sources. Generated Python code runs in a sandbox, and tool calls are capped to limit security and resource risks. Like other branch-based methods, CBPO samples several continuations and therefore incurs additional generation cost. A fixed rollout budget bounds this cost, and all methods are compared using matched rollout slots. The experiments evaluate benchmark accuracy rather than deployment safety. High-stakes applications would require domain-specific testing, content filtering, access controls, and human oversight.

References

[1] P. Agrawal, A. Samanta, S. Ghasemlou, B. Vidolov, J. Bhandari, K. Asadi, D. Jiang, and A. Modi. 2026. Off-Context GRPO: Learning to Reason on Hard Problems using Privileged Information. arXiv preprint arXiv:2607.19313. doi:10.48550/arXiv.2607.19313

[2] D. Cheng, S. Huang, X. Zhu, B. Dai, W. X. Zhao, Z. Zhang, and F. Wei. 2025. Reasoning with Exploration: An Entropy Perspective. arXiv preprint arXiv:2506.14758. doi:10.48550/arXiv.2506.14758

[3] K. Cobbe et al. 2021. Training Verifiers to Solve Math Word Problems. arXiv preprint arXiv:2110.14168. doi:10.48550/arXiv.2110.14168

[4] G. Dong et al. 2025. Agentic Reinforced Policy Optimization. arXiv preprint arXiv:2507.19849. doi:10.48550/arXiv.2507.19849

[5] G. Dong et al. 2025. Tool-Star: Empowering LLM-Brained Multi-Tool Reasoner via Reinforcement Learning. arXiv preprint arXiv:2505.16410. doi:10.48550/arXiv.2505.16410

[6] G. Dong, L. Bao, Z. Wang, K. Zhao, X. Li, J. Jin, J. Yang, H. Mao, F. Zhang, K. Gai, G. Zhou, Y. Zhu, J.-R. Wen, and Z. Dou. 2025. Agentic Entropy-Balanced Policy Optimization. arXiv preprint arXiv:2510.14545. doi:10.48550/arXiv.2510.14545

[7] J. Feng et al. 2026. ReTool: Reinforcement Learning for Strategic Tool Use in LLMs. In *International Conference on Learning Representations*.

[8] L. Feng, Z. Xue, T. Liu, and B. An. 2025. Group-in-Group Policy Optimization for LLM Agent Training. arXiv preprint arXiv:2505.10978. doi:10.48550/arXiv.2505.10978

[9] Z. Gou et al. 2024. ToRA: A Tool-Integrated Reasoning Agent for Mathematical Problem Solving. In *International Conference on Learning Representations*.

[10] D. Guo et al. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv preprint arXiv:2501.12948. doi:10.48550/arXiv.2501.12948

[11] Y. He, H. Wu, S. Liu, H. Ge, H. Zhou, K. Wu, Z. Zheng, Q. Lin, Z. Zhong, and Y. Zhang. 2026. Rethinking Token-Level Credit Assignment in RLVR: A Polarity-Entropy Analysis. arXiv preprint arXiv:2604.11056. doi:10.48550/arXiv.2604.11056

[12] D. Hendrycks et al. 2021. Measuring Mathematical Problem Solving with the MATH Dataset. arXiv preprint arXiv:2103.03874. doi:10.48550/arXiv.2103.03874

[13] Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing A Multi-hop QA Dataset for Comprehensive Evaluation of Reasoning Steps. In *Proceedings of the 28th International Conference on Computational Linguistics*. International Committee on Computational Linguistics, Barcelona, Spain (Online), 6609–6625. doi:10.18653/v1/2020.coling-main.580

[14] Z. Hou, Z. Hu, Y. Li, R. Lu, J. Tang, and Y. Dong. 2025. TreeRL: LLM Reinforcement Learning with On-Policy Tree Search. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

[15] J. Hu et al. 2025. REINFORCE++: Stabilizing Critic-Free Policy Optimization with Global Advantage Normalization. arXiv preprint arXiv:2501.03262. doi:10.48550/arXiv.2501.03262

[16] Y. Ji, Z. Ma, Y. Wang, G. Chen, X. Chu, and L. Wu. 2026. Tree Search for LLM Agent Reinforcement Learning. In *International Conference on Learning Representations*.

[17] Chuang Jiang, Mingyue Cheng, Xiaoyu Tao, Qingyang Mao, Jie Ouyang, and Qi Liu. 2026. TableMind: An Autonomous Programmatic Agent for Tool-Augmented Table Reasoning. In *Proceedings of the Nineteenth ACM International Conference on Web Search and Data Mining*. ACM, Boise, ID, USA, 260–270. doi:10.1145/3773966.3777932

[18] B. Jin, H. Zeng, Z. Yue, J. Yoon, S. O. Arik, D. Wang, H. Zamani, and J. Han. 2025. Search-R1: Training LLMs to Reason and Leverage Search Engines with Reinforcement Learning. In *Conference on Language Modeling*.

[19] X. Lai et al. 2024. Step-DPO: Step-Wise Preference Optimization for Long-Chain Reasoning of LLMs. arXiv preprint arXiv:2406.18629. doi:10.48550/arXiv.2406.18629

[20] X. Li, H. Zou, and P. Liu. 2025. ToRL: Scaling Tool-Integrated Reinforcement Learning. arXiv preprint arXiv:2503.23383. doi:10.48550/arXiv.2503.23383

[21] Y. Li et al. 2026. Rethinking On-Policy Distillation of Large Language Models: Phenomenology, Mechanism, and Recipe. arXiv preprint arXiv:2604.13016. doi:10.48550/arXiv.2604.13016

[22] H. Lightman et al. 2024. Let's Verify Step by Step. In *International Conference on Learning Representations*.

[23] H. Lin and Z. Xu. 2025. Understanding Tool-Integrated Reasoning. arXiv preprint arXiv:2508.19201. doi:10.48550/arXiv.2508.19201

[24] Z. Lin, T. Liang, J. Xu, X. Wang, R. Luo, C. Shi, S. Li, Y. Yang, and Z. Tu. 2024. Critical Tokens Matter: Token-Level Contrastive Estimation Enhances LLM's Reasoning Capability. arXiv preprint arXiv:2411.19943. doi:10.48550/arXiv.2411.19943

[25] Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. 2023. Measuring and Narrowing the Compositionality Gap in Language Models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*. Association for Computational Linguistics, Singapore, 5687–5711. doi:10.18653/v1/2023.findings-emnlp.378

[26] R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, S. Ermon, and C. Finn. 2023. Direct Preference Optimization: Your Language Model Is Secretly a Reward Model. In *Advances in Neural Information Processing Systems*.

[27] Mandeep Rathee, Sean MacAvaney, and Avishek Anand. 2025. Quam: Adaptive Retrieval through Query Affinity Modelling. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining*. ACM, Hannover, Germany, 954–962. doi:10.1145/3701551.3703584

[28] J. Schulman et al. 2017. Proximal Policy Optimization Algorithms. arXiv preprint arXiv:1707.06347. doi:10.48550/arXiv.1707.06347

[29] Z. Shao et al. 2024. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. arXiv preprint arXiv:2402.03300. doi:10.48550/arXiv.2402.03300

[30] H. Sun, Z. Qiao, J. Guo, X. Fan, Y. Hou, Y. Jiang, P. Xie, Y. Zhang, F. Huang, and J. Zhou. 2025. ZeroSearch: Incentivize the Search Capability of LLMs without Searching. arXiv preprint arXiv:2505.04588. doi:10.48550/arXiv.2505.04588

[31] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multi-hop Questions via Single-hop Question Composition. *Transactions of the Association for Computational Linguistics* 10 (2022), 539–554. doi:10.1162/tacl_a_00475

[32] S. Wang et al. 2025. Beyond the 80/20 Rule: High-Entropy Minority Tokens Drive Effective Reinforcement Learning for LLM Reasoning. arXiv preprint arXiv:2506.01939. doi:10.48550/arXiv.2506.01939

[33] J. Wei et al. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems*.

[34] Jialong Wu, Wenbiao Yin, Yong Jiang, Zhenglin Wang, Zekun Xi, Runnan Fang, Linhai Zhang, Yulan He, Deyu Zhou, Pengjun Xie, and Fei Huang. 2025. WebWalker: Benchmarking LLMs in Web Traversal. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vienna, Austria, 10290–10305. doi:10.18653/v1/2025.acl-long.508

[35] A. Yang et al. 2025. Qwen3 Technical Report. arXiv preprint arXiv:2505.09388. doi:10.48550/arXiv.2505.09388

[36] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, 2369–2380. doi:10.18653/v1/D18-1259

[37] S. Yao et al. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations*.

[38] Q. Yu et al. 2025. DAPO: An Open-Source LLM Reinforcement Learning System at Scale. arXiv preprint arXiv:2503.14476. doi:10.48550/arXiv.2503.14476

[39] Jie Zhang, Dongsheng Bi, Tao Sun, Minghui Yang, Jian Wang, and Yiwei Wang. 2026. TOOL-CURE: Tool Selection via Curriculum-Enhanced Reinforcement Learning with Sample Screening for LLMs. In *Proceedings of the Nineteenth ACM International Conference on Web Search and Data Mining*. ACM, Boise, ID, USA, 946–954. doi:10.1145/3773966.3777952

[40] Yue Zhang, Shicheng Zhou, Xiaopeng Li, Zhiliang Tian, Yifu Gao, Shiyi Zhang, Wenqing Hou, Yuying Liu, and Bin Zhou. 2026. KnowFC: Navigating Knowledge Conflicts in Large Language Model-based Fact-Checking. In *Proceedings of the Nineteenth ACM International Conference on Web Search and Data Mining*. ACM, Boise, ID, USA, 996–1006. doi:10.1145/3773966.3777935

[41] C. Zheng, S. Liu, M. Li, X.-H. Chen, B. Yu, C. Gao, K. Dang, Y. Liu, R. Men, A. Yang, J. Zhou, and J. Lin. 2025. Group Sequence Policy Optimization. arXiv preprint arXiv:2507.18071. doi:10.48550/arXiv.2507.18071